102(b) because the reference was not published until the Compcon '97 conference held February 23-26, 1997. (See, IEEE/IEE Electronic Library Online citation and abstract attached with the paper.) Since the author Rodney Limprecht is named as an inventor on the present application, the paper also fails to qualify under § 102(a). MPEP § 706.02a states, "for 35 U.S.C. 102(a) to apply, the reference must have a publication date earlier in time than the effective filing date of the application, and must not be applicant's own work." (See also, MPEP § 716.10, which states the following example 1: "During the search the examiner finds a reference fully describing the claimed invention. The applicant is the author or patentee and it was published or patented less than one year prior to the filing date of the application. The reference cannot be used against applicant since it does not satisfy the 1-year time requirement of 35 U.S.C. 102(b).") No other subsection of 35 U.S.C. § 102 applies.

The claims therefore clearly are allowable over the cited art.

Patentability Over Background Art And Schwartz

Claims 1-4 have been rejected under 35 U.S.C. § 103(a) as unpatentable over Background-described art in view of Schwartz. Applicants traverse the rejection.

Claim 1

Claim 1 is generally directed to enhancing the scalability of server applications through components of the server application having control over the duration of their state in memory. More specifically, the claim recites,

> 1. In a computer having a main memory, a method of enhancing scalability of server applications, comprising:
>
> executing an application component under control of an operating service, the application component having a state and function code for performing work responsive to method invocations from a client, the application component maintaining the state in the main memory between the method invocations of the function code by the client; and

destroying the state by the operating service in response to an indication from the application component to the operating service that the work is complete and without action by the client.
(Emphasis added.)

The Office asserts that the recited "executing" and "destroying" actions are taught by the Background-described art, with the exception that the recited "destroying" is "without action by the client." The Office further asserts that modifying the Background-described art to perform the recited "destroying... without action by the client" is suggested by Schwartz teaching "managing resources (client/server connections), wherein the lifetime of a resource (connection) can be... controlled by the server without action by the client (server breaks the connection) after the work for the client is completed." Applicants strongly disagree.

First, Schwartz fails to teach that the server disrupts the connection "without action by the client" and fails to teach the disruption occurs "in response to an indication from the application component that the work is complete." Schwartz at column 20, line 60 through column 21, line 42 details the actions that occur as part of the server disrupting a connection. These actions include a number of actions at the client described at column 21, lines 17-42. Accordingly, Schwartz fails to teach the server disrupts the connection "without action by the client."

Schwartz in the Abstract states, "after the communication has been completed, the connection can be disconnected by the client or broken by the server." However, Schwartz lacks any teaching that the server's disruption is "in response to an indication ... that the work is complete." There is no statement in Schwartz that the connection produce any indication which prompts the server to disrupt the connection at completion of the communication or any other time. Schwartz merely states that the connection can be broken by the server sometime "after the communication has been completed."

The Background-described art also lacks any teaching or suggestion that the component's state is destroyed without action by the client, or that the component provide any indication to an operating service that its processing work is complete.

Schwartz and the Background-described art (taken either individually or in combination) therefore lack both these limitations of the claim.

Further, Applicants have added the language, "the application component maintaining the state in the main memory between the method invocations of the function code by the client." This language is not met by the server applications programmed as described at page 2, line 10 through page 3, line 2 (i.e., server-side "stored procedures"). As indicated at page 2, line 18, such procedures store any state acquired during an invocation away to secondary storage upon completing processing each "single interaction." The object-oriented programming techniques described at page 3, line 3 through page 5, line 4 is clearly stated in the Background And Summary Of The Invention section at page 3, line 5 to be "antithetical" to the "stored procedures" programming approach. Accordingly, a programming model that has both the storing away of state to secondary memory at the completion of processing an operation, and storing state in main memory between method invocations clearly is not taught or suggested by the Background-described art.

Schwartz and the Background-described art therefore clearly also lack the recited "application component maintaining the state in the main memory between the method invocations of the function code by the client."

Moreover, a motivation to combine these references is clearly lacking. As noted by the Office, the described object-oriented programming is stated in the Specification at page 4, lines 12-13, as having the object "kept in memory until the client's reference to the object is released." This statement would clearly lead away from any modification where an object is released while a client retains a reference to the object. Further, this asserted modification would alter the principle of operation of the described object-oriented programming model where

"so long as any client programs have a reference to an object's instance, data associated with the instance is maintained in memory to avoid the client issuing a call to an invalid memory reference" as stated in the Specification at page 4, lines 4-6. (See, MPEP § 2143.01, which states, "the proposed modification cannot change the principle of operation of a reference.") Accordingly, Schwartz simply would not have led one or ordinary skill in the art to make the asserted modification to the "stored procedures" or "object-oriented programming model" described in the Specification.

For these reasons, claim 1 (and its dependent claims 2-4) clearly are allowable over this art.

Claim 2

Claim 2 depends from claim 1, and further recites, "wherein the operating service retains an operating service's reference to the application component and the step of destroying the state comprises releasing the operating service's reference to the application component by the operating service while the client retains a client's reference to the application component." This further recitation is not taught or suggested by the cited art.

The Office asserts that it would be obvious to apply the technique of "in object-oriented programming, destroying an object's state is typically achieved by releasing reference to the object." Applicants have amended claim 2 to more clearly recite that the operating service releases its reference to the object while the client retains its reference. Per object oriented programming, the object's state would not be destroyed under these circumstances. (See, Specification at page 3, line 26 through page 4, line 13.)

For this additional reason, claim 2 is separately patentable over this art.

Claim 3

Claim 3 depends from claim 1, and further recites, "wherein the step of destroying the state comprises resetting the state of the application component to

the application component's initial post-creation state (emphasis added)." The Office asserts that the Background-described art "teaches resetting the state." Not so. As described in the Specification at page 2, lines 26-27, applications programmed as stored procedures load user's state into main memory at the start of a procedure. This does not destroy the state of the procedure, and is not performed under the circumstances for the "destroying" recited in claim 1. Also, there is no teaching or suggestion to destroy the state of an application component by resetting its state.

Nevertheless, Applicants have clarified claim 3 by adding the language, resetting "to the application component's initial post-creation state." There is no teaching or suggestion to destroy an application component's state by resetting to its initial post-creation state.

For this additional reason, claim 3 is separately patentable over this art.

Claim 4

Claim 4 depends from claim 1, and further recites, "said destroying the state is performed by the operating service upon a next return of the application component from the client's call following the indication from the application component that the work is complete." The Office asserts it would have been obvious to apply the teaching of a C++ destructor. Applicants respectfully submit that the C++ destructor does not operate in the manner claimed in claim 4. Specifically, the C++ destructor is called at the times described at page 868 in Adams et al., "C++, An Introduction To Computing" (1995) (a copy of which is submitted herewith), which include at the end of a client's "main function," "block," or "function." This return at the end of the client's function is not the next return of the object (it's the return of the client), and does not follow the object making an indication to the operating service that its processing work for a client is complete. C++ therefore also fails to teach or suggest this claim limitation.

For this reason, claim 4 is separately allowable over the cited art.

CONCLUSION

The claims in their present form should now be allowable. Such action is respectfully requested.

Respectfully submitted,

KLARQUIST SPARKMAN CAMPBELL
LEIGH & WHINSTON, LLP

By _____
Stephen A. Wight
Registration No. 37759

One World Trade Center, Suite 1600
121 S.W. Salmon Street
Portland, Oregon 97204
Telephone: (503) 226-7391
Facsimile: (503) 228-9446

cc: Patent Group Docket Dept. (80683.1USORG)
KS Paralegal